12 | 九条性能测试的经验和教训：如何保证测试结果可靠且可重复？

2019-12-23 庄振运

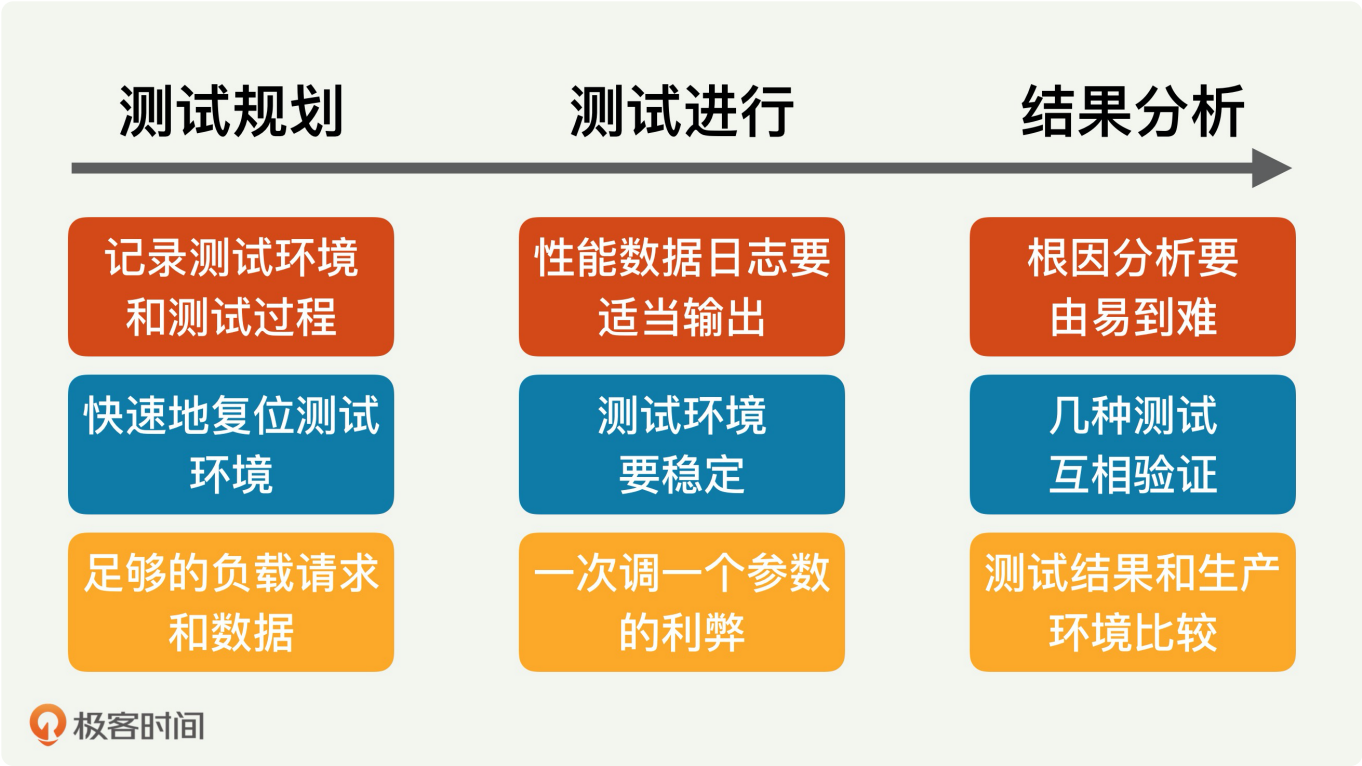
性能工程高手课 进入课程 >

讲述：庄振运
时长 15:35　大小 14.28M

你好，我是庄振运。

上一讲我们介绍了十几种常用的性能测试工具。我们知道，性能测试的一个关键是保证测试结果可靠、可重复，否则就没有意义。所以，我们今天来学习一下进行性能测试时，这方面的经验和教训。

根据以前做过的相关工作，我总结了九条这样的经验和教训。按照逻辑时间顺序，我将它们大体上分成三大类别，就是测试前的规划、测试中的变化和测试后的结果分析；每一类又有三条要点。



测试规划

三大类别的第一类别是测试规划，我们先来说说测试规划时的要注意的三条要点。

1. 详细记录测试环境和测试过程

做每个性能测试时，测试的环境至关重要。这里的环境包括软件硬件、操作系统、测试的负载、使用的数据等等。

测试的环境不同，性能测试的结果可能会迥异。除了测试环境，其它几个因素比如测试的过程，包括步骤和配置的改变也有相似的重要性。所以，我们每次测试都要把测试环境和测试过程记录下来，为将来分析数据做参考。

这些测试环境信息包括什么呢？大体上是操作系统和程序的版本号，以及各种软件参数的设置等等。

记录测试环境的目的是为了以后的各种分析。比如我们如果发现两次测试结果不匹配，需要找到不匹配的原因，那么这些测试环境就是相当关键的信息。如果两次测试结果的不同是因为软件配置不同导致的，根据记录的测试环境信息，我们就很容易根因出来。

至于如何记录，我们可以去手工去记录，但最好是自动记录，因为手工记录既费时费力，又容易出错。如果我们知道常用的环境配置的路径，可以很方便的写个程序来自动记录。或者

依靠能自动记录配置的软件也可以。

2. 快速地复位测试环境

有时候性能测试需要重复进行多次，那么就需要在每一次测试后，能够有快速“复位”到初始测试环境的机制。这个复位机制越简单有效越好，最好能达到所谓“一键复位”的程度，从而最大限度地降低手工复位的工作量。

复位的具体内容和方式要根据性能测试的情况而定，这就要分析一下在每个测试过程中，哪些子系统的配置和参数被改动或者影响了。

虽然理论上说，我们只需要恢复那些会影响测试结果的配置和参数，但是由于系统的复杂性，有些乍看起来不会影响测试结果的参数，其实或多或少也会影响测试结果。所以一般来讲，我们会**把所有的配置和参数都恢复一下**。环境复位的具体方式包括重新拷贝文件、重置配置参数、清空各种缓存等。

举个具体例子，如果是测试一个存储系统的性能，比如硬盘的性能。如果测试过程中写入了大量数据到存储系统，那么测试完毕，恢复环境的时候，就需要删掉这些数据，以便使被测系统回到初始状态。

再举个类似的例子，如果是测试一个数据库的性能，那么测试过程可能发出了很多查询请求。因为数据库经常会缓存各种情况和结果，所以在恢复测试环境时，不要忘记把缓存也清空。

一个现代计算机系统中往往有很多种缓存，比如数据库缓存、文件系统缓存、存储缓存（比如 SAN）等。不同种类的缓存自然有不同的清空方式。对于数据库缓存，在每个测试之前可以用命令行来刷新数据库缓存；如果数据库不提供这样的命令，则需要重新启动数据库。对于文件系统缓存，一般需要重新启动服务器。对于 SAN 缓存，可以考虑在测试期间减少甚至关闭缓存，或者用大量的随机数据来“污染”缓存。

3. 足够的负载请求和数据

性能测试需要流量负载以及相关的数据，我们需要特别注意保证它们的多样化和代表性。否则测试结果会严重失真。

当使用相同的测试数据进行重复测试时，如果负载请求不够大，那么各种缓存可能会严重影响结果。

如何识别缓存的影响呢？

各级缓存系统都有相应的统计指标和命令，比如文件系统缓存和 SAN 缓存中的缓存命中率，就可以通过统计信息中报告的延迟，并且结合经验来识别。举例来说，如果一个随机 8KB 或 16KB 数据的对硬盘的读写，测量出的延迟不到 1 毫秒，那就实在是“太快”了，快得让人不敢相信；可以肯定它是命中某种缓存了。

除了合理清空缓存外，更有效地方式是保证测试时间足够长、测试的负载请求足够多和数据足够多样化，从而最大限度地减少或者掩盖缓存等其他因素的影响。

测试进行

测试规划之后，我们就要关注测试中的变化了。

1. 性能数据日志要适当输出

性能测试过程中，也需要实时输出有关的性能数据和日志，比如 CPU 使用率数据。这些数据对于测试完成后的分析至关重要。

输出的数据和日志最好保存起来，以方便后期处理 / 重新处理。比如执行过很多不同参数配置的测试之后，我们经常需要进行测试之间的比较，这时候就需要仔细检查以前输出的日志了。

输出数据的多少也需要注意，虽然我们希望尽量多地输出，但是也要意识到，太多的输出有时候会起反作用。

1. 存储的开销：可以用压缩存储来解决。
2. 数据处理时间的开销：注意压缩和解压缩也要花时间。
3. 可能影响性能测试的结果：有时候因为某些原因，日志输出会影响被测系统的性能，比如往一个文件写入日志，可能会导致系统的暂停，从而影响测试的结果（这个我们下一讲会详细剖析）。

2. 测试环境要稳定

性能测试的环境在测试进行过程中，以及重复测试时一定要保持稳定和一致，否则测试结果就不可靠或者不能重复。

比如一个测试进行中，背景流量的负载产生了剧烈变化，导致被测系统的延迟增加。假如这个背景流量变化不是预期产生在测试规划之内的，就会造成测试结果失真。

类似的，如果测试过程中有不可控的因素，造成每次重复测试结果都不同，那这样的测试就不可靠。无论测试结果好坏，都不能用来作出有用的结论。

一般的解决方案，是尽量在一个独立无干扰的环境中进行测试，加上每次测试都准确地恢复测试环境，就能最大限度地保证测试环境的稳定。

3. 一次调一个参数的利弊

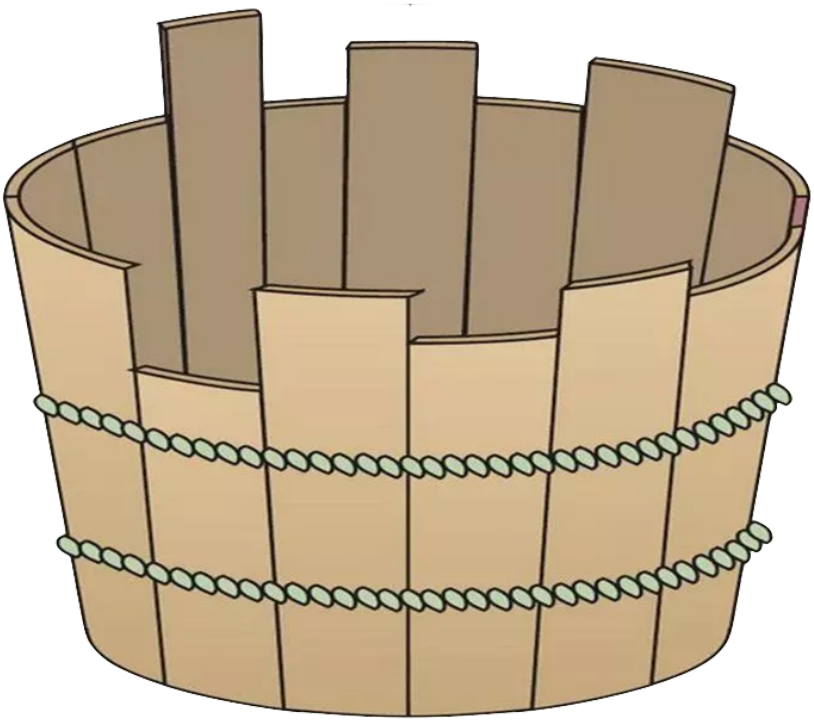
在性能调优测试中，就是通过实验来找出系统的最优配置。

对这种测试，我们经常听到的经验是，“一次只调一个参数，通过对比实验，就能知道这个参数的最佳值“。你觉得这个经验对吗？

我们先看这个经验的出发点。性能调优过程中有很多可调参数和配置，互相之间的影响不清楚，因此不宜对系统的各种参数进行随意的改动。应该以基本参考设置为基础，逐次根据实际测试结果进行优化，一次只对某个领域进行性能调优，并且每次只改动一个设置和参数，避免其他参数和相关因素的干扰。

这个经验有它的道理，但我觉得这样做既对也不对，你不能盲从，否则就会错过最优配置的机会。

我举个生活中的例子来说明。假设有一个装水的木桶，这个木桶由多块可调整高度的木条组成。假设这些木条的初始高度不一，我们的目的是找到一个木条高度组合，从而实现最大的装水量。这个问题看起来很简单，根据木桶定律，就是把每块木条都调到最高嘛。



我们假设这个木桶是我们的被测系统，每块木条就是一个参数。再假设我们对木条之间的关系不清楚。如果一次只调整一个参数，然后实验测试装水量。因为木桶装水量取决于高度最低的木条，我们或许会得出结论——只有那块最低木条值得调，其他木条的高度都不重要。

这种结论的结果就是把最低木条调高，比如调到最高。那么这个所谓的“最优系统”，也就是整个木桶的装水量取决于新的最低木条，也就是原来木桶中高度次低的那个木条。

我们很容易看出，如果一次调整多个木条，那么我们就会很快地找出一个更优的系统，也就是一个装水更多的木桶。

结果分析

测试后的结果分析也是需要你关注的重点，三条经验如下：

1. 根因分析要由易到难

如果在性能测试过程中需要查找性能瓶颈，查找的过程一定要由易到难逐步排查。因为参考我们学过的帕累托法则，从最明显的性能瓶颈来开始，往往可以事半功倍。

首先从最常见的几种资源和几个指标查起，比如 CPU 使用率、存储 IO 繁忙度、内存大小、网络发送和接收速度等。

进一步的分析就可以针对不太明显的资源，比如内存带宽，缓存击中率，线程加锁解锁等；从而过渡到应用程序和系统的一些配置参数。这些配置参数包括应用服务器及中间件，操作系统瓶颈，数据库、WEB 服务器的配置；还有应用业务瓶颈，比如 SQL 语句、数据库设计、业务逻辑、算法、数据等。

2. 几种测试最好互相验证

各种性能测试工具和测试手段都有自己的局限性或缺陷，从而可能会造成测试结果出现偏差。所以，如果条件和时间允许，最好使用几种不同测试工具或手段，分别进行独立的进行测试，并将结果相互比较和验证。

如果几种测试比较后结果相似，那么皆大欢喜。

否则就需要进行深入比较分析，弄明白造成结果不同的原因。分析以后，如果能够清楚地了解根因并作出合理解释，那么很多时候就够了，可以止于此。最后形成结论时，只要稍加有针对性地说明就可以。

3. 测试结果和生产环境比较

如果性能测试是在非生产环境中进行的，那么得出的测试结果或许会和生产环境大相径庭。如果我们测试的目的是尽量和生产环境一致，就需要仔细审查每个测试的环节，包括测试环境和测试流程。

假如非生产环境的测试结果和生产环境的测量不同，很多情况下是由于测试环境不同导致，尤其要注意的是网络环境的差异。

比如测试的环境是在局域网，而真正的生产环境是无线网或者 4G，那么可以肯定，这样的局域网测试没有什么意义。网络差异的影响经常会大大超过很多人的预期，因为实际的生产环境还牵扯很多上层协议和背景流量，比如 HTTP 协议。

假设 4G 用户在网络层或者链路层上，相对局域网而言只是多了 30 毫秒的延迟。在 Web 服务器中等负载的情况下，这几十毫秒的链路层延迟，就可能导致应用层响应时间增加惊人

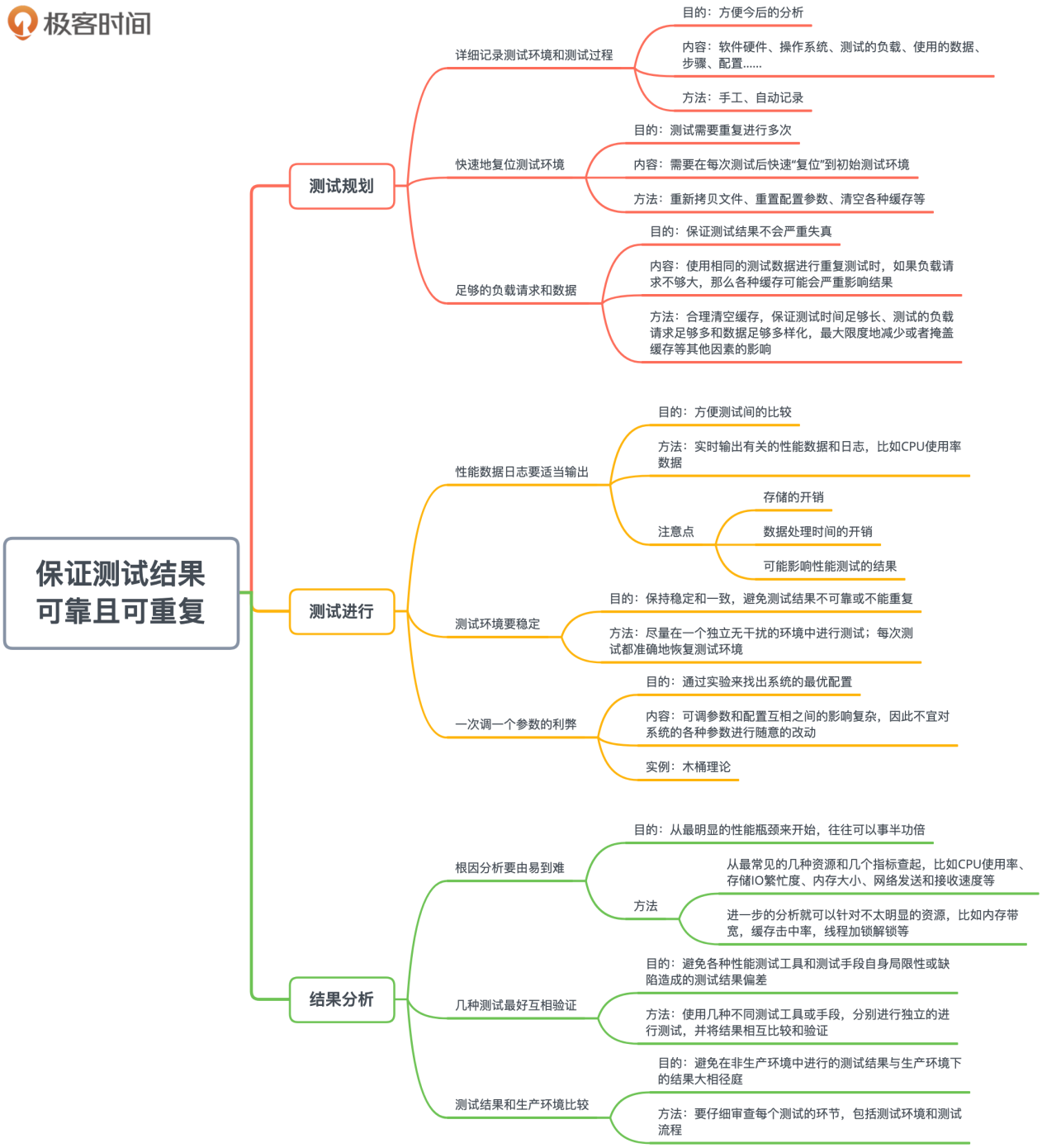
的几十秒延迟。

对网络环境这点，最理想的解决方式当然是在生产环境进行测试。如果实现这点有困难，那么可以考虑使用一定的网络仿真，来模拟真实生产环境。

除了网络环境的影响，真实客户的操作流程也可能造成测量结果的不同。比如一个真正的 Web 用户或者 APP 用户在访问我们的系统时，往往会有思考或者阅读时间；所以真实世界的用户，不太会在 1 秒钟内发出背对背的好几个页面请求。对于这种情况的差异，可以通过人为地添加停顿来模拟思考时间。

现在很多测试工具都支持这种思考时间的引入。比如使用 JMeter，可以使用高斯随机计时器（Gaussian Random Timer），来模拟现实世界中的用户，以随机方式和页面进行交互。

总结



当年毛主席给彭德怀写过几句诗：

山高路远坑深，

大军纵横驰奔。

谁敢横刀立马？

唯我彭大将军。

打仗有很多坑，做性能工作也有很多坑。我们要把性能测试的工作做好，本身就不容易。山高路远都不在乎，但一定要注意其中的陷阱。

所以只有不断地学习别人总结的经验，吸取别人的教训，解决测试的各种挑战，才能得出可靠，可重复并且有意义的性能测试结果。

思考题

回想一下你做过的性能测试，有没有踩过今天介绍的几个坑？

踩过之后，是怎么防止以后不再踩同样的坑的？

对没有踩过的坑，你有没有方法避免以后中招呢？

欢迎你在留言区分享自己的思考，与我和其他同学一起讨论，也欢迎你把文章分享给自己的朋友。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　11 | 性能测试的工具：七大测试场景如何选择高质量的测试工具？

精选留言 (1)

写留言

Linuxer

2019-12-23

今天文章中提到的坑基本都踩过，针对之前踩过的坑，我们采取了两种办法避免重复踩坑，一种是能自动化检测的自动化检测，另一种是形成一种流程规范，针对流程中的每个点都有一些检查项，根据检查项提前检查来规避之前遇到过的问题。
没踩过的坑，这个点就说不准了。

展开